A Temporal Data Model for Multimedia Database Systems

Ilan Eini[¶], Vera Goebel[§] and Bjørn Skjellaug[¶]

[¶] Department of Informatics, University of Oslo, PB 1080, 0316 Oslo, Norway, {ilane, bjornsk}@ifi.uio.no
[§] UNIK, University of Oslo, PB 70, 2007 Kjeller, Norway, goebel@unik.no

Research Report 252
ISBN 82-7368-172-6
Department of Informatics
University of Oslo
June 1997

Abstract: The presented data model is a novel approach for integrating temporal concepts into a multimedia database system. Multimedia objects are extended with the traditional time dimensions valid time and transaction time. In addition a new time dimension specifically tailored for multimedia data types is presented with semantics that are completely orthogonal to the already established time dimensions, valid time and transaction time, i.e., the model supports a 3D time for multimedia data. This new time dimension, the play time dimension, places the building blocks of multimedia data in a temporal structure for multimedia presentation. This model is currently being implemented in a MMDBS for distance education at UNIK, University of Oslo.

1. Introduction

The most common time dimensions used in temporal database systems are valid time and transaction time. The **valid time** of a fact is the time when the fact was, is or will be true in the modeled reality and the **transaction time** of a fact is when a fact is current in the database and may be retrieved or modified [4]. Transaction time is managed solely by the database system. The multimedia data types that are dependent on time for data presentation have their own presentation time dimension, coined here the **play time** dimension. The distance education project at UNIK is a multimedia application that is composed of many different multimedia data types. This application greatly benefits from a formal temporal structuring of the multimedia data types since many of these data types depend on clear synchronization semantics in the data model. Another advantage is that temporal data types and multimedia data types can be handled by the same database system and their temporal aspects can be managed in a uniform manner [7].

2. Play time Dimension

This dimension's time-values position each Logical Data Unit (LDU)[1] relative to the media data type's start-time. The play time dimension is not anchored in time to a global clock like the valid time or transaction time dimensions.

A temporal attribute, is formally defined as a pair (t, f(t)), where t is a time instant and f is a partial function such that for all instants t where f is defined, f(t) denotes the attribute value at time t. The prerequisite for this definition is that the t values are fixed in time. The time-values in the play time dimension are not fixed in time until the user starts or alters the presentation of a media object. Additionally, unlike the valid time and transaction time dimensions, the play time

[1] The application and context decide what a logical data unit is. In this context, we choose it to be the smallest data unit that is not decomposable into units of shorter duration. This can be a frame for a video object or a sample for an audio object. For a more in depth discussion of LDU, see [9].

dimension is bi-directional since the presentation can also be performed backwards. Streams and Computer Generated Multimedia data (CGM) have different temporal characteristics. We will consequently look at their play time dimension separately.

2.1 Play time Dimension for Streams

The LDUs in a stream are a set of data units that are ordered according to the normal playback mode, and they all represent an equal duration of time. A stream has LDUs of fixed duration such as an audio sample or a video frame. We start with the definitions concerning a stream S:

*Definition 1: A **finite stream sequence S** is defined as follows:*

$$S = \{LDU_0, LDU_1, LDU_2, \ldots\ldots, LDU_{n-1}\}$$

where S(i) returns LDU_i for $0 \le i \le n-1$. ■

*Definition 2: Let S be a finite stream sequence, then the **cardinality of S**, i.e., the number of LDUs in S, is given by |S|.* ■

*Definition 3: Let S be a sequence of LDUs then the **ordering of the LDUs in S** is given by the relation $<_S$ such that $LDU_i <_S LDU_j$ iff $i<j$* ■

The moment the user initiates the playback of a multimedia object, the time values t_i ($t_i \in \mathcal{TIME}$, where $0 \le i \le n$ and $\mathcal{TIME}$ is the play time dimension) are bound to the actual time dimension, e.g., a clock. Thus, in normal playback mode LDU_i is played back during the interval $[t_i, t_{i+1}\rangle$. In reverse playback mode LDU_i is played back during the interval $\langle t_{i+1}, t_i]$.

A *chronon* is a non-decomposable time-interval of some fixed minimal duration, and is the smallest time-unit addressable by the application. A chronon can be set to whatever an application needs [4], e.g., millisecond, second, hour, year etc. Let t_i and $t_{i+1} \in \mathcal{TIME}$, then the duration $[t_i, t_{i+1}\rangle$ is a chronon. A LDU_duration is the time it takes to display one LDU.

*Definition 4: A **LDU_duration** is the play time dimension's equivalent to a chronon.* ■

The speed of the presentation is then given by the ratio:

$$\text{Presentation\_speed} = \frac{(\text{presentation LDU\_duration})}{(\text{original LDU\_duration})}$$

In addition to the relation $<_S$ from Definition 3 that describes the ordering in a stream we need a relation to describe the ordering of LDUs during presentation. This ordering is identical to $<_S$ during playback in forward direction. During playback in reverse direction it is the inverse of $<_S$. But first we need to define what a presentation is:

*Definition 5: A **stream presentation SP** is defined as follows: **SP**=<**Sr**, **pf**, **pl**, **pd**, **pspeed**>, where **Sr** is a reference to a stream S, **pf** and **pl** are the first and last index numbers of the LDUs of S to be presented respectively, **pd** is the presentation direction and **pspeed** is the presentation speed.* ■

A stream presentation may additionally contain parameters to describe other aspects of the presentation that are media data type specific. A stream presentation as defined in Definition 5 is contiguous. To create a presentation with gaps and changes in temporal characteristics such as speed or direction, one needs a collection of several stream presentations with temporal relationships between them as described in [2].

*Definition 6: The **presentation ordering**[2] $<_P$ of a stream presentation SP is defined as follows: Let's say that **Sr** references the stream S, the presentation direction variable **pd** may assume either the value* `forward` *or the value* `backward` *for the entire **Sr**-interval, and that* $0 \le \boldsymbol{pf} \le \boldsymbol{pl} \le |S|-1$ *then*

$$\forall i,j \in \{\mathbf{pf},...,\mathbf{pl}\}\big(\forall \mathbf{pd} \in \{\,forward, backward\}$$
$$\big[(i < j \wedge \mathbf{pd} = forward) \vee (j < i \wedge \mathbf{pd} = backward) \Leftrightarrow (LDU_i <_P LDU_j)\big]\big) \blacksquare$$

As a consequence of Definition 6 we observe that the following also must hold:

$$\forall i,j \in \{\mathbf{pf},...,\mathbf{pl}\}\big(\forall \mathbf{pd} \in \{\,forward, backward\}$$
$$\big[(i < j \wedge \mathbf{pd} = forward) \Leftrightarrow (LDU_i <_S LDU_j) \wedge (t_i < t_j) \wedge (LDU_i <_P LDU_j)\big] \wedge$$
$$\big[(i < j \wedge \mathbf{pd} = backward) \Leftrightarrow (LDU_i <_S LDU_j) \wedge (t_j < t_i) \wedge (LDU_j <_P LDU_i)\big]\big)$$

In the data model, we distinguish between the logical data model describing how the data is actually stored and the presentation model describing how the data should be presented. We can informally say that $<_S$ is used for the ordering of LDUs in the logical data model and $<_P$ is used for ordering LDUs in the presentation model and the observation above describes the relationship between $<_S$ and $<_P$.
Both valid time and transaction time can be added to a play time dependent data type, since they are orthogonal to the play time dimension. Example 1 illustrates this point.

Example 1: Assume for instance that a surveillance camera of a bank only records when it registers movement. The time-values of the play time dimension are assigned to every recorded frame from the start of the recording and stop when the recording does. Later whenever the video media object is played back, the time-values are bound to a physical time-line.
The time-values of the valid time dimension of the video sequence registers the time when the movement actually occurred. The transaction time of the video sequences tells us when the video was stored and available for querying in the database. These times may not be the same if the video is committed to the database at a later time than it was recorded.

2.2 Play time Dimension for Computer Generated Multimedia Data.

By Computer Generated Multimedia data (CGM) we mean multimedia data types that are specifications of commands and operations that a computer must interpret and execute. More intuitive examples of CGMs are animations, music, and speech. But recordings of user input such as keyboard- and mouse-commands can also be CGMs. CGMs and streams differ as follows:

1. A CGM is a specification of operations that are executed at certain points in time according to a schedule, while streams are sets of LDUs that are interpreted and presented at regular intervals.
2. The time scale of a CGM is more elastic. It can be manipulated more freely than the time scale for streams [2].
3. Events can be instantaneous[3] or they can have duration in which to execute. Consequently, events need another type of timestamping than regular LDUs in streams.
4. Events can execute both in sequence and in parallel for a single multimedia data type, while LDUs for a single stream are always displayed sequentially.

[2] The relation $<_P$ is read like this: "is presented before"
[3] In practice, all events have a certain duration, but if this duration is smaller than the granularity of the given time scale used in the specification, and is not intended to have a certain duration, then we say that it is instantaneous.

We assume an event as the basic building block for CGM. An event has start-time, duration and stop-time. An event can either be executed once or periodically at regular intervals. When a CGM is produced all its temporal specifications are according to a specific time-scale. A major goal with this data model is to combine streams and CGMs in a presentation in a uniform and flexible way. Thus, we must conform the time scale used in the temporal specifications of a CGM to the play time dimension. One granule in that time scale corresponds to a chronon, or LDU_duration in Definition 4.

3. Data Model

For temporal database management systems (TDMBS) there are many suggestions on how to realize temporal capabilities in the data model [8][10]. Multimedia data models such as SGML/HyTime in [11] and Mediadoc in [5] are more concerned with the multimedia data types and do not have the precise semantics of time like the pure temporal data models have. In [9], a number of scheduling and synchronization techniques are introduced for authoring a multimedia presentation such as the Interval-based, Axes-based, Control flow-based and Event-based. The model presented in this paper combines these techniques and draw upon their advantages. The syntax is based on the ODMG Object Definition Language (ODL)[4]. The presentation of multimedia data can be done in many different ways. Hence it is beneficial to separate the *logical data model* of multimedia data from the *presentation model* which is somewhat like a "view" of a traditional database. This is also done in the SGML/HyTime data model [11] through their event schedule.

In our model, we are using an object-oriented approach so each element is identified by a unique object identity. Objects may be complex with attributes that are objects on their own. We present the logical data model as a type hierarchy to offer the users the opportunity to extend the model with new multimedia data types in every level of abstraction.

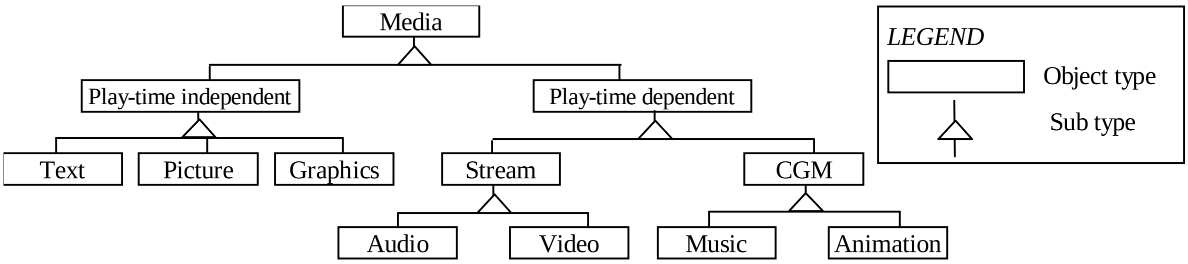

Figure 1: The logical data model type hierarchy

Figure 2 illustrates parts of the data model implemented in ODL. The timestamp object type is supposed to be subtyped by an object type that has time-values from the time dimensions that are desired. The interface `Temporal` stems from [7]'s idea for extending basic data types with temporal capabilities. Figure 2 also describes the structure of a video object. It consists of a global specification of properties for all its frames and the stream object type contains its LDUs. The `Virtual_video` interface is taken from the presentation model and describes how an interval of a video object should be presented. A video presentation may consist of several `Virtual_video` objects. Each leaf node in Figure 1 has a corresponding object type in the presentation model that describes how it should be presented.

[4] Draft of the ODMG 2.0 standard available at the ODMG official Web site located at http://www.odmg.org on April 15. 1997.

```
interface Temporal{
     attribute Set<struct<Timestamp      T,
                             Media        M >> M_history
};

interface Timestamp{};

interface Stream_timestamp:Timestamp{
  attribute Play_time pt;
  attribute Valid_time vt;
  attribute Transaction_time tt_start,tt_stop;
};

interface LDU:Media{
     ..............
};

interface Frame:LDU{
  attribute Bit_String Frame_data;
};

interface Video:Stream{
     attribute Temporal m_data;*
     attribute double LDU_duration;*
     attribute Compression_scheme cs;
     attribute Resolution res;
     attribute Color_depth cd;
     .................
};

interface Virtual_video:Virtual_Stream{
     attribute Media Sr; *
     attribute Play_time pf, pl; *
     attribute double pspeed; *
     attribute enum pd {forward, backward}; *
     attribute boolean color;
     attribute Resolution presentation_res;
     attribute int skip_frames;
     attribute set<effect> effects;*
     ..............
};
```

Figure 2: An abstract of the ODL specification of some of the object types in the data model.[5]

The components in the presentation model work as *virtual multimedia objects* and a composite multimedia object can reference these objects just as if they were real media objects. In a distributed environment it may be necessary to present multimedia data with lower Quality-of-Service (QoS) than what is available because of system limitations. Thus, the object types in the presentation model have several parameters that consider QoS. In Figure 2 `Virtual_video` has the parameters `skip_frames`[6], `color` and `presentation_res` which may override the default presentation settings so that various QoS-settings are possible. The QoS concern extends throughout the presentation model. Effects that are executable in real-time can also be added to a presentation object. These effects reference specific intervals of the presentation object.

4. Conclusions

In this paper we have introduced a new temporal dimension, the play time dimension, and briefly showed how it can be applied to an object-oriented data model. The integration of a multimedia database with a temporal data model and query languages, extended with a play time dimension, is especially useful for achieving a uniform management of multimedia data and other temporal

---

[5] The attributes marked with * are originally inherited (from super types omitted here because of space limitations), but are included here for illustration purposes.

[6] Skip_frames indicates that the presentation should only display every nth frame.

data, i.e., multimedia data associated with 3D time values is accessed by means of the generic temporal language constructs.

The data model outlined is currently being implemented in the object-oriented database system ObjectStore and applied to the electronic classroom application [2]. The data from the lectures are stored in the database using the data model presented in this paper. After a lecture is recorded the lecturer can manipulate the data and created several alternate presentations of the same lecture, for instance with different languages and different QoS parameters. These presentations also contain temporal relationships that utilize the semantics of the play time dimension.

Acknowledgment: The authors would like to thank Ragnar Normann for most useful suggestions to an earlier version of this paper. This research was in part supported by the Norwegian Research Council through grant MOI.31297.

5. References

[1] E. Bertino, E. Ferrari, G. Guerrini, A formal Temporal Object-Oriented Data Model, In P. Apers, editor, Proc. Fifth International Conference on Extending Database Technology, Avignon (France), March 1996.

[2] V. Goebel, T. Plageman, I. Eini, P. Halvorsen, K. Lund, H.C. Løchsen, L. Rønning, Distributed Multimedia System Activities at UNIK – The Electronic Classroom Scenario – , Technical Report, UNIK, University of Oslo, June 1997.

[3] I. Goralwalla, Y. Lentiev, M. T. Özsu and D. Szafron, A Uniform Behavioral Temporal Object Model, University of Alberta, TR 95-13, July 1995.

[4] Christian S. Jensen et al., A Glossary of Temporal Database Concepts, ACM SIGMOD Records, 23(1): 52–64, March 1994.

[5] Ahmed Karmouch, James Emery, A Playback Schedule Model for Multimedia Documents, IEEE MultiMedia Vol. 3, No. 1, Spring 1996.

[6] Y. Masunaga, An Object-Oriented Apporach to Temporal Multimedia Data Modeling, IEICE Transactions on Information and Systems, Vol E78-D, No. 11, November 1995.

[7] B. Skjellaug, A-J. Berre, A Uniform Temporal Geographic Data Model Framework, Technical Report STF40A97046, SINTEF Telecom and Informatics, Oslo, 1997.

[8] B. Skjellaug, Temporal Data: Time and Object Databases, Research Report 245, Dep. Of Informatics, Univ. of Oslo, 1997.

[9] R. Steinmetz, K. Nahrstedt, Multimedia: Computing, Communications & Applications, Prentice Hall, 1995.

[10] A.U. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, R.T. Snodgrass, Temporal databases: Theory, Design and Implementation, Benjamin/Cummings, Redwoo Cty CA, 1993.

[11] C. Vittal, M.T. Özsu, D. Szafron, G. El-Medani, The Logical Design of a Multimedia Database for a News-on-Demand Application., Technical Report TR94-16, Dep. of Computing Science, Univ. of Alberta, Edmonton, Canada, 1994.